ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступень вищої освіти бакалавр

на тему «**Розробка програмного забезпечення щодо обліку наукової та професійної діяльності викладачів мовою програмування C#»**

Виконав: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Ніколайчук Максим Анатолійович
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

КИЇВ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність підготовки 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В.Негоденко
“____” _____ 2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУЮ РОБОТУ СТУДЕНТУ

Ніколайчуку Максиму Анатолійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи: «Розробка програмного забезпечення щодо обліку наукової та професійної діяльності викладачів мовою програмування C#»

Керівник роботи Гаманюк Ігор Михайлович, с.в.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «12» березня 2021 року №65

2. Строк подання студентом роботи «1» червня 2021року

3. Вихідні дані до роботи: Локальний застосунок для обліку наукової та професійної діяльності викладачів, методи та засоби проєктування та розробки програмного забезпечення

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1 Огляд предметної області
4.2 Вибір засобів реалізації
4.3Проєктування застосунку
4.4 Реалізація застосунку

5. Перелік графічного матеріалу (презентація)
1.

2.
3.
4.

6. Дата видачі завдання__1.06.2021__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської Роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на бакалаврську роботу	19.04.21	
2	Огляд предметної області	20.04.21	
3	Вибір інструментальних засобів реалізації	23.04.21	
4	Проєктування застосунку	24.04.21	
5	Реалізація веб-застосунку	28.04.21	
6	Тестування веб-застосунку	03.05.21	
7	Написання та оформлення пояснювальної записки	10.05.21	
8	Розробка графічних та презентаційних матеріалів	15.05.21	
9	Захист бакалаврської роботи	01.06.21	
10			

Студент _____ М.А.Ніколайчук
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.М. Гаманюк
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 41 с., 31 рис., 8 джерел.

1 КЛЮЧОВІ СЛОВА: C# , .NET.CORE

2 ОБ’ЄКТ ДОСЛІДЖЕННЯ: ОБЛІК НАУКОВОЇ ТА ПРОФЕСІЙНОЇ ДІЯЛЬНОСТІ ВИКЛАДАЧІВ

Предмет дослідження:Програмне забезпечення щодо обліку наукової та професійної діяльності викладачів

Мета роботи:Розробка програми та поліпшення облікової системи наукової та професійної діяльності викладачів

Методи дослідження: Уніфікований процес розробки програмного забезпечення.

У роботі було :

Проведено аналіз існуючих рішень розробки додатку, їх переваги та недоліки. Обраний оптимальний засіб розробки мобільного додатку, а саме фреймворк для C# .NET.

Були визначені вимоги та функціональні можливості додатку “Lecturer Online”.

Був проведений аналіз програмних засобів для .NET. Обраний оптимальний стек технологій, який забезпечує швидку розробку, та перевикористання коду.

Результат даної роботи полягає в можливості впровадження цього додатку в життя викладачів університетів. Особливість даного сервісу це можливість обліку наукової та професійної діяльності викладачів.

ЗМІСТ

3 ВСТУП

У міру розвитку людства,її потреби множились і зростали різноманітними шляхами , а бажання людей задовольнити стільки потреб, скільки вони могли б на максимально високому рівні якості та якомога швидше породив безпрецедентний процес. Наука та знання еволюціонували швидкими темпами саме для того, щоб вони могли не відставати від потреб та бажань людства, що спричиняє постійно зростаюче споживання ресурсів,практично забуваючи, що ресурси обмежуються потенціалом планети. Ігноруючи той факт, що людство втратило ресурси, а напруга між потребами і ресурсами не покращились, а натомість збільшились і розширились. Усі ці зміни
не зробили нас щасливішими за наших прабабусь і дідусів, яких
володіють, але елементарними засобами задоволення своїх потреб. Що змінило спосіб нашого досвіду змін? Прогрес людства - безпосередньо прив'язаний до технічного, технологічного та інформаційного прогресу,
їх бажання якомога краще і якомога швидше задовольнити свої потреби на високому рівні, а чому б і ні, задовольнити їхні потреби теж на гідному рівні. Збільшення потреб , пов'язане з інновацією і технологічними змінами, тобто вкладки допоміжних процесів в Інтернеті та зміни в процес виробництва, на якому їх відносила ця відносно нова технологія.
Тепер, коли Інтернет розроблений, ми створили для себе нову
соціальну структуру, і з цього ми намагаємось створити нові структури в економічній, соціальний, політичний та рівні безпеки. Інтернет та телекомунікаційні системи представляють основу цієї нової соціальної структури.

Об’єкт дослідження – облік наукової та професійної діяльності викладачів

Предмет дослідження – програмне забезпечення щодо обліку наукової та професійної діяльності викладачів.

Мета роботи – розробка програми та поліпшення облікової системи наукової та професійної діяльності викладачів.

Методи дослідження – уніфікований процес розробки програмного забезпечення.

Наукова новизна даної роботи полягає у створенні першої облікової програми наукової та професійної діяльності викладачів.

1 МЕТОДИ РОЗРОБКИ ЛОКАЛЬНИХ ЗАСТОСУНКІВ

1.1Комп'ютерна розробка на сьогоднішній час

Комп'ютери проникли в усі сфери діяльності людини, починаючи з початкової освіти і закінчуючи вивченням новітніх технологій, вивчення нових видів матерії, невідомих поки людству. Застосування комп'ютерних технологій полегшує процес освіти в середніх і вищих навчальних закладах як самих учнів, студентів, так і робочого персоналу.Завдяки різноманітності програмного і апаратного забезпечення сьогодні можливе використання всіх потенційних можливостей комп'ютерних технологій. Це дозволяє зберігати величезну кількість інформації, займаючи при цьому мінімальне місце. Також комп'ютерні технології дозволяють швидко цю інформацію обробляти і тримати її в захищеному вигляді.Широке поширення ПК зіграло величезну роль у розвитку ринку праці. Автоматизація обробки інформації дозволяє в лічені секунди виконати роботу, на яку раніше губилися тижні, інформування керівників про стан підприємств і робочих місць відбувається миттєво. Збільшується економічний потенціал в галузі страхових і фінансових послуг завдяки збільшеному обміну послуг. Впровадження комп'ютерних технологій для введення нових форм зайнятості та організації праці.На розробку нових проектів витрачається набагато менше часу, бо не треба витрачати багато часу на обчислювальні процеси і можна повністю присвятити час самого процесу. Велику роль комп'ютерні технології відіграють в медицині, створюються різні віртуальні моделі розвитку захворювань, створюються величезні бази інформації на підставі яких знаходяться нові препарати для лікування.Комп'ютер сьогодні є засобом для спілкування, а сама зв'язок на даний момент найдешевша. Для людей з обмеженими можливостями часом це єдиний спосіб не тільки спілкування, а й завдяки сучасним комп'ютерним технологіям такі люди можуть себе реалізувати, отримати роботу.Комп'ютерні технології надають позитивних ефект в розвитку дітей при правильному їх використанні. Помічено, що при грамотному підборі програм та ігор у дітей

краще розвивається логічне мислення, поліпшується координація очей і рук. У дитини розвивається самовпевненість і почуття власної гідності, діти більш зосереджені в порівнянні з дітьми, які не мають досвіду користування комп'ютером.З іншого боку необмежений доступ до величезних обсягів інформації іноді призводить до надмірного використання комп'ютера, в основному це Інтернет-залежність або залежність від комп'ютерних ігор. А це завдає як психологічний, так і фізичний шкоду. Люди надмірно захоплені комп'ютерними іграми більш дратівливі, запальні в звичайному спілкуванні. У деяких розвивається залежність від ігор, і при неможливості задовольнити свою потребу в звичайному світі погіршується настрій, з'являються стану підвищеної тривожності, іноді депресії.

1.2Історія появи комп'ютерної розробки

Перші мови програмування з'явилися задовго до появи перших комп'ютерів. Ще в 19-му столітті існували “програмовані ” ткацькі верстати та піаніно-програвачу , спосіб програмування яких нагадує так звані предметно-орієнтовані мови програмування. На початку 20-го століття починають використовувати перфокарти та механічну обробку даних. В 1930-1940 рр. виникає лямбда-числення та машина Тьюринга , які застосовували математичну абстракцію для опису алгоритмів. Лямбда-числення згодом здійснило вплив на проектування мов програмування.

Комп’ютери почали свою діяльність у 1822 р. Винаходом диференціального двигуна Чарльзом Беббіджем. Це був фактично перший механічний комп'ютер в історії; він міг виконувати прості обчислення і виконувати ці завдання, змінюючи передачі для різних операцій. . «Найбільш ранньою формою комп’ютерної мови був фізичний рух».

У 1945 році Джон фон Нойман, який працював в Інституті перспективних досліджень, розробив дві важливі концепції, які допомогли сформувати майбутнє комп’ютерного програмування. Першим пунктом, який він зробив, став відомий як "Техніка спільної програми", де говорилося, що власне обладнання комп'ютера має бути простим і не вимагати ручного підключення

до кожної програми. Натомість слід використовувати складні інструкції для управління простим обладнанням, що дозволяє перепрограмувати час пізніше набагато швидше. Друга його ідея, відома як "умовна передача контролю", визнала той факт, що хронологічні комп'ютерні інструкції повинні бути замінені блоками коду, до яких можна отримати доступ у будь-якому порядку в будь-який час. Друга половина цієї ідеї викладає ідею логічних тверджень, таких як оператор If then та циклічні оператори.

У 1957 році світ вперше спробував на смак основну мову програмування у вигляді FORTRAN (його назва означає систему перекладу FORmula). Ця мова була розроблена IBM для наукових розрахунків, і хоча вона добре працювала, вона не була розроблена для обробки вхідних даних та результатів користувачів, як очікувало більшість людей / компаній.

3.2.1 1.3.Що таке мова програмування C#?

C # є "матір'ю всіх мов програмування" і є найвидатнішою мовою програмування. Якщо ви хочете ступити ногою вперед, щоб завоювати сферу програмування, ви потрапили в потрібне місце. Для початківців програмістів C # - найкраща мова для початку.

C # - одна з найбільш примітивних мов, оскільки вона тісно пов'язана з мовами низького рівня. Загальновідомим є той факт, що C # - це мова програмування високого рівня, яка лежить у кінцевому спектрі низького рівня мови високого рівня. Отже, міцний фундамент в C # є обов’язковим, якщо ви хочете розвивати свою кар’єру в програмуванні.

Це основа для вивчення вирішення проблем та програмування, оскільки вона передбачає розробку логістичного підходу до вирішення найосновніших проблем, з якими ви стикалися у своєму підручнику з математики в початковій школі чи щоденних реальних ситуаціях, що вимагають вирішення конкретного алгоритму .

C# є процедурною мовою програмування ,а також загальною мовою програмування,розробленою Деннісом Річі в 1972 році.Це дивовижна і проста мова ,яка допомагає легко розробляти складні програмні

додатки.Вважається рідною мовою всіх мов.С - мова програмування високого рівня ,яка також забезпечує підтримку мови програмування низького рівня .С складається з ряду понять, починаючи від змінних,функцій,операторів,обсягу тощо.

Процедурна мова слідую добре організований архітектурі,показуючи всі кроки ,необхідні компьютеру ,щоб отримати бажаний результат.Мови програмування ,такі як С,Fortran,BASIC,Pascal та С++,є процедурним програмуванням.

Що розуміється під мовою програмування високого та низького рівня?

Мова високого рівня схожа на людську ,її легко розуміти та писати.Мова високого рівня більше фокусується на арифметичних операціях,ефективності програми та простоті кодування.Іншими словами ,мова програмування високого рівня на відміну від машинного рівня .Це тісно пов'язане з мовою ,якою ми розмовляємо ,тобто людською мовою.Ми використовуємо його при легкому та зрозумілому розробленні програми

Мова низького рівня - це дуже зручна для машин мова.Тому писати програми мовою низького рівня дуже важко для людини.

3.5.1.1 1.4.Особливості програмування на C#

Існують різні особливості, або ми можемо сказати причини вивчення програмування на C#, які роблять його популярним у технічній та управлінській галузях. Основними рисами мови C# є:

Простий та ефективний - стиль синтаксису легко зрозуміти. Ми можемо використовувати C# для розробки додатків, які раніше були розроблені мовою збірки.

Керування пам'яттю - це дозволяє розподіляти пам’ять під час виконання, тобто підтримує концепцію динамічного розподілу пам’яті.

Динамічне розподіл пам'яті - коли ви не впевнені у вимогах до пам'яті у вашій програмі і хочете вказати її під час запуску, тобто під час запуску програми, ви можете зробити це вручну.

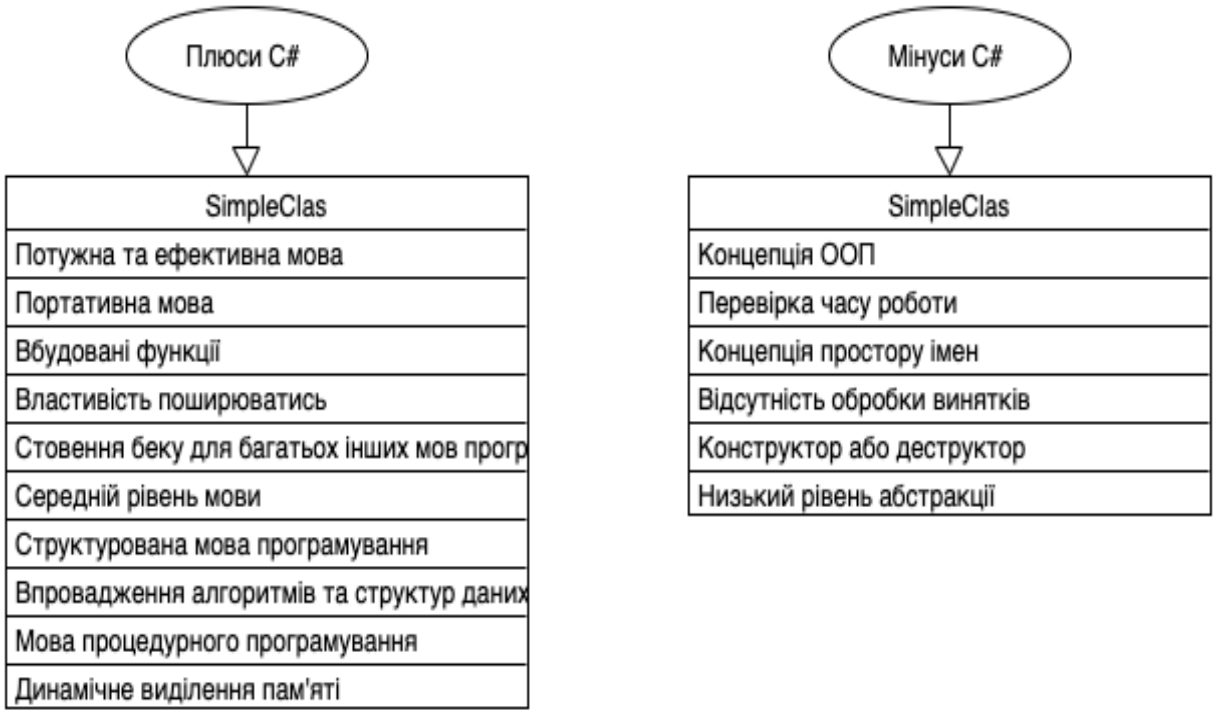
Покажчики - мова C# забезпечує вказівник, який зберігає адресу пам'яті як своє значення. Покажчики корисні для зберігання та доступу до даних із пам'яті. Ми детально вивчимо це в наших майбутніх підручниках.

З урахуванням регістру - Цілком зрозуміло, що малі та великі регістри розглядаються по-різному в C. Це означає, що якщо ви пишете „program” і „Program”, обидва вони означатимуть різні значення в C#. має нижній регістр, тоді як "P" у програмі має верхній регістр.

На основі компілятора - C є мовою на основі компілятора, тобто для виконання коду нам спочатку потрібно скомпілювати його.

Структурно-орієнтований / модульний - C# - це структурована мова програмування. Це означає, що ви можете розділити свій код та завдання всередині функції, щоб зробити її інтерактивною. Ці функції також допомагають у повторному використанні коду

3.5.1.2 1.5.Плюси та мінуси програмування на C#



1.5.1.Плюси та мінуси написання коду на C

Плюси :

1. Потужна та ефективна мова:
   C# - це надійна мова, оскільки вона містить безліч типів даних та операторів, що дає вам широку платформу для виконання всіх видів операцій.
2. Портативна мова:
   C дуже гнучкий, або ми можемо сказати незалежний від машини, який допомагає запускати ваш код на будь-якій машині, не вносячи жодних змін або лише декілька змін у код.

3. Вбудовані функції:
   У ANSI C є лише 32 ключові слова, що мають багато вбудованих функцій. Ці функції корисні при побудові програми на C#.
4. Властивість поширюватися:
   Інша вирішальна здатність C# - поширюватися. Ми вже вивчали, що мова C# має власний набір функцій у бібліотеці C#. Таким чином, стає легко використовувати ці функції. Ми можемо додати наші власні функції до стандартної бібліотеки C# та спростити код.
5. Створення беку для багатьох інших мов програмування
   C вважається найбільш фундаментальною мовою, яку потрібно вивчати, якщо ви починаєте з будь-якої мови програмування. Багато мов програмування, таких як Python, C ++, Java та ін., Побудовані на основі мови C#.
6. Структурована мова програмування:
   C# заснований на структурі. Це означає, що проблеми або складні проблеми поділяються на менші блоки або функції. Ця модульна структура допомагає спростити та спростити тестування та обслуговування.
7. Середній рівень мови:
   C# - мова програмування середнього рівня, що означає, що вона підтримує програмування високого рівня, а також програмування низького рівня. Він підтримує використання ядер та драйверів у програмуванні низького рівня, а також підтримує системні програмні програми мовою програмування високого рівня.
8. Впровадження алгоритмів та структур даних:
   Використання алгоритмів та структур даних в C# зробило обчислення програм дуже швидкими та плавними. Таким чином, мова C# може використовуватися в складних обчисленнях та операціях, таких як MATLAB.
9. Мова процедурного програмування:
   C# дотримується відповідної процедури щодо своїх функцій та підпрограм. Оскільки воно використовує процедурне програмування, C# стає простіше ідентифікувати структуру коду та вирішити будь-яку проблему в певній серії коду. У процедурному програмуванні змінні та функції оголошуються перед використанням.
10. Динамічне виділення пам'яті:

С забезпечує динамічне розподіл пам'яті, що означає, що ви можете розподілити пам'ять під час роботи. Наприклад, якщо ви не знаєте, скільки пам’яті потрібно об’єктам у вашій програмі, ви все одно можете запустити програму на С та одночасно призначити пам’ять.

Мінуси:

1. Концепція ООП:
   C# - дуже обширна мова, але вона не підтримує концепцію ООП (спадкування, поліморфізм, інкапсуляція, абстракція, приховування даних). C# просто дотримується підходу процедурного програмування.
2. Перевірка часу роботи:
   У мові програмування C# помилки чи помилки не виявляються після кожного рядка коду. Натомість компілятор показує всі помилки після написання програми. Це робить перевірку коду дуже складною у великих програмах.
3. Концепція простору імен:
   С не реалізує концепцію просторів імен. Простір імен структурований як ланцюжок команд, що дозволяє повторно використовувати імена в різних контекстах. Без просторів імен ми не можемо оголосити дві однакові змінні.Але в програмі C# цієї функції бракує, а отже, ви не можете визначити змінну з тим самим іменем у C#.
4. Відсутність обробки винятків:
   Обробка винятків - одна з найважливіших особливостей мов програмування. Під час компіляції коду можуть виникати різні аномалії та помилки. Обробка винятків дозволяє виявити помилку та прийняти відповідні відповіді. Однак C# не виявляє цієї важливої особливості.
5. Конструктор або деструктор:
   С немає конструктора або деструктора.
   Конструктори та деструктори підтримують основні функції об’єктно-орієнтованого програмування. Обидві - це функції-члени, які створюються, як тільки створюється об'єкт класу. Пізніше ви будете детально вивчати конструктор та деструктор.
6. Низький рівень абстракції

C# - невелика та основна машинна мова, яка має мінімальне приховування даних та ексклюзивну видимість, що впливає на безпеку цієї мови.

3.5.2

Висновки

З урахуванням того, що C# вирішує більшість відомих мінусів кросс-платформних додатків, для розробки локального застосування буде використовувати саме цю технологію.

C# дає нам можливість значною мірою перевикористати код при розробці аналогічного додатка для інших платформ.

4 ІДЕЯ ДОДАТКУ LECTURER ONLINE

Відповідно до вимог держави завданням вищих навчальних закладів є підготовка конкурентоздатного фахівця ,якісними характеристиками якого є :вміння йти на розумний ризик ,потреба в автономії, схильність до творчості ,саморозвиток у наумоємному освітньому просторі.

В розробці даного додатку лежить дві ідеї:

Перша з них це спрощення пошуку інформації про викладачів для їх керівників.

Друга ідея заключається в можливості абітурієнтів переглядати інформацію про викладачів університету в який вони збираються подавати документи.

2.1.Основні можливості мобільного-застосунку «Lecturer Online»

В кожного викладача додатку “Lecturer Online” є власний куточок з повною інформацією про його наукову та професійну діяльності .У додатку є можливість дивитится, додавати, видаляти а також змінювати інформацію про викладачів їх керівниками.

Висновки

Зараз немає простого та зручного інструменту для обліку наукової та професійної діяльності викладачів отже Lecturer Online виступить як розв'язання цієї проблеми. Все тому що Lecturer Online має легкий та зрозумілий дизайн та не потребує жодних навичок.

3 ВИБІР СТЕКУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ

[redacted]Вибір середовища розробки

Для написання коду на C# є багато середовищ розробки однак найпопулярніші це :

Середовища розробки
1.Visual Studio Code
2.Eclipse
3.NetBeans
4.Sublime Text
5.Atom
6.CodeLite
7.CodeWarrior
8.MonoDevelop

3.1 Середовища розробки на мові C#

1.Visual Studio Code - Це редактор коду з відкритим кодом, розроблений Microsoft для Windows, Linux та Mac OS. Visual Studio Code заснований на фреймворці Electron. Згідно з опитуванням, проведеним у 2018 році компанією Stack Overflow, він був визнаний найбільш популярним інструментом середовища для розробників серед інших. Крім того, цей IDE також настроюється, що дозволяє програмістам змінювати тему, ярлики ключових слів та налаштування.
Основні переваги:

- Підтримка налагодження
- Підсвічування синтаксису
- Інтелектуальне заповнення коду, фрагменти та переробка коду
- EmbeddedGit Control
- Повністю портативний
- Проста настройка

2.Eclipse - Це одна з найпопулярніших, потужних і корисних середовищ розробки, яка використовується розробниками для програмування на C / C ++. Це програмне забезпечення з відкритим кодом, яке є простим і простим у використанні. Спочатку він використовувався для програмування на Java, але зараз використовується для різних мов. Eclipse може працювати під управлінням Windows, Linux та Mac OS. Ви навіть можете подати помилку на їх веб-сайті, якщо ви зіткнетеся з такою в IDE Eclipse або компіляторі.

Основні переваги:

- Чудовий графічний інтерфейс користувача з функцією перетягування
- Підтримує статичний аналіз коду
- Смарт-заповнення коду
- Підсилювачі продуктивності
- Інтеграція Git
- Підтримка між платформами
- Багата спільнота

3.NetBeans-Це одна з найбільш часто використовуваних IDE і може працювати на Windows, Linux, Mac OS X та Solaris. Це безкоштовна IDE із відкритим кодом, написана на Java. Ця IDE складається з інтерфейсу, який постачається з функцією перетягування та переліку зручних шаблонів проектів. Ви можете використовувати NetBeans для створення програм C / C ++ з динамічними та статичними бібліотеками. Це дозволяє програмістам створювати програми C / C ++ із існуючого коду. Крім того, він надає великий набір інструментів для програмістів на C / C ++. NetBeans має безліч плагінів, які можуть розширити програмне забезпечення. Розробники також можуть віддалено контролювати розвиток свого проекту.

Основні переваги:

- Підтримка між платформами
- Багатий набір плагінів
- Підтримує кілька мов програмування
- Просте та ефективне управління проектами
- Інтелектуальне редагування коду
- Велика спільнота підтримки

4.Sublime Text-Це крос-платформний редактор вихідного коду, який підтримує кілька мов та мов розмітки. Sublime Text має інтерфейс програмування на програмі Python і має елегантний користувальницький інтерфейс, а також неймовірні функції та дивовижну продуктивність. Програмісти також можуть додавати додаткові функції за допомогою численних плагінів, створених спільнотою.

Основні переваги:

- Перейти до будь-чого - Швидка навігація до символів, рядків або слів

- Перейти до визначення - Може автоматично генерувати загальний проектний індекс кожного класу, методу та функції.
- Дозволяє кілька варіантів вибору
- Палітра команд
- Потужний API
- Дуже настроюється
- Міжплатформна підтримка Mac, Windows та Linux
- Підсвічування синтаксису
- Автозавершення
- Плагіни та інтеграції

5.Atom-Це один із висококонфігуруваних текстових редакторів, який є безкоштовним для особистих та комерційних розробок. Він був розроблений і розроблений GitHub і має дуже велику спільноту. Для цього редактора доступна безліч плагінів, що розширює його функції та робить його більш настроюваним. Він також підтримує OS X, Windows та Linux і має дуже простий інтерфейс для роботи.
Основні переваги:

- Підтримка між платформами
- Розумне автозавершення
- Менеджер пакетів
- Підтримка плагінів
- Кілька вікон
- Можливість пошуку та заміни тексту
- Підтримує палітру команд

6.CodeLite-Ще одна хороша IDE для програмування на C або C ++, яку використовує безліч програмістів. Це програмне забезпечення з відкритим кодом, яке може працювати на всіх основних платформах, включаючи Windows, Linux та OS X. Він забезпечує чудову підтримку компіляторів, а також дозволяє користувачам дізнатися більше про помилки, просто натиснувши на нього.
Основні переваги:

- Перевірка правопису
- Завершення слів

- Двигун завершення коду на основі Clang
- Завершення коду JavaScript, PHP
- Графічна різниця утиліта
- Підтримка Valgrind
- Плагіни Git та Svn
- Провідник баз даних

7.CodeWarrior-Це повна IDE, відома тим, що забезпечує надзвичайно візуальну та автоматизовану структуру для швидкої розробки програми. CodeWarrior був опублікований NXP Semiconductors для редагування, компіляції та налагодження програмного забезпечення. Цей IDE може працювати в ОС Windows та Linux і може спростити найскладніші дії, роблячи роботу розробника простою та легшою.

Основні переваги:

- Простий у використанні графічний інтерфейс користувача
- Дозволяє єдиний шлюз до всіх компонентів
- Керівник проекту
- Забезпечує єдиний шлюз для всіх компонентів
- Високо оптимізується
- Потужний макроасемблер
- Забезпечує швидкий доступ до різних елементів вихідного коду, таких як змінні, класи та інші
- Розумний лінкер
- Програмісти можуть створювати власні бібліотеки
- Кольоровий багатошаровий редактор із синтаксичним спрямуванням

8.MonoDevelop-Це текстовий редактор, в якому ви легко пишете настільні та веб-програми на Linux, Windows та Mac OS X. За допомогою MonoDevelop можна також перенести .NET-програми, створені за допомогою Visual Studio, на Linux та Mac OS X.

Основні переваги:

- Багатоплатформна
- Розширене редагування тексту
- Підтримка декількох мов

- Інтегрований налагоджувач
- Настроюваний робочий стіл
- Створюйте веб-проекти з повним завершенням коду

Подивившись на найпопулярніші середовища розробки було обрано Visual Studio. Воно дає всі необхідні можливості для створення додатку.

Вибір фреймворку

Для створення додатку був великий вибір фреймворків однак виділимо найпопулярніші:

Фреймворки
1.ASP.NET Web Forms
2.IdentityServer
3.ASP.NET Web API
4.ASP.NET AJAX
5..NET
6.ASP.NET Core
7.Entity Framework

3.2. Фреймворки C#

1.ASP.NET.Web Forms - це веб-програма і одна з декількох моделей програмування, що підтримуються технологією Microsoft ASP.NET.

2.IdentityServer - це безкоштовний фреймворк OpenId Connect та OAuth 2.0 для ASP.NET Core з відкритим кодом.

3. ASP.NET Web API - це розширювана структура для створення служб, заснованих на HTTP, до яких можна отримати доступ у різних додатках на різних платформах, таких як Інтернет, Windows, мобільні пристрої тощо.

4.ASP.NET AJAX-який раніше називався Atlas, - це набір розширень для ASP.NET, розроблений корпорацією Майкрософт для реалізації функціональних можливостей Ajax. Він випускається під загальнодоступною ліцензією Microsoft (Ms-PL).

5..NET-Програмне забезпечення, яке працює в основному на Microsoft Windows. Він забезпечує контрольоване середовище програмування, де програмне забезпечення можна розробляти, встановлювати та виконувати в операційних системах на базі Windows.

6.ASP.NET Core-Міжплатформна високопродуктивна платформа з відкритим кодом для створення сучасних хмарних програм, підключених до Інтернету. Програми ASP.NET Core можуть працювати на .NET Core або на повній .NET Framework.

7.Entity Framework-Фреймворк об’єктів / реляційного картографування. Це вдосконалення ADO.NET, яке надає розробникам автоматизований механізм доступу та зберігання даних у базі даних. Він був частиною .NET Framework, але з версії Entity framework 6 він відокремлений від .NET Framework.

Подивившись на найпопулярніші фреймворки було обрано ASP.NET Core. Воно дає всі необхідні можливості для створення додатку

3.3 Використання інструментів для контролю версій Git та GitLab

Системи контролю версій дозволяють декільком розробникам, дизайнерам і членам команди працювати разом над одним проєктом. Ці системи мають вирішальне значення для забезпечення доступу кожного до останньої версії коду.

Git - найбільш часто використовувана система контролю версій. Git відстежує зміни, які ви вносите у файли, тому у вас є запис про те, що було зроблено, і ви можете повернутися до певних версіями, якщо вам коли-небудь знадобиться[14]

GitLab - це сервіс, схожий на GitHub, який забезпечує внутрішнє веб управління DevOps репозиторіями Git. GitLab пропонує два варіанти: безплатну версію для спільноти та платну корпоративну версію. GitLab міститись безліч функцій, необхідних для успішного управління процесом розробки програмного забезпечення, таких як Jira integration, CI runner, release management, binary attachments, та багато іншого.

3.4.Висновки

В даному розділі проведений аналіз та обрані інструменти для роботи .Було обрано середовище розробки Visual Studio Code.Був обраний фреймворк .NET.Core. Був обраний архітектурний підхід до реалізації . Для контролю версії, та зберігання коду буде використана GitLab. На реалізацію цього додатку було

обрано оптимальний стек технологій, який забезпечує швидку розробку та комфортний і простий функціонал .

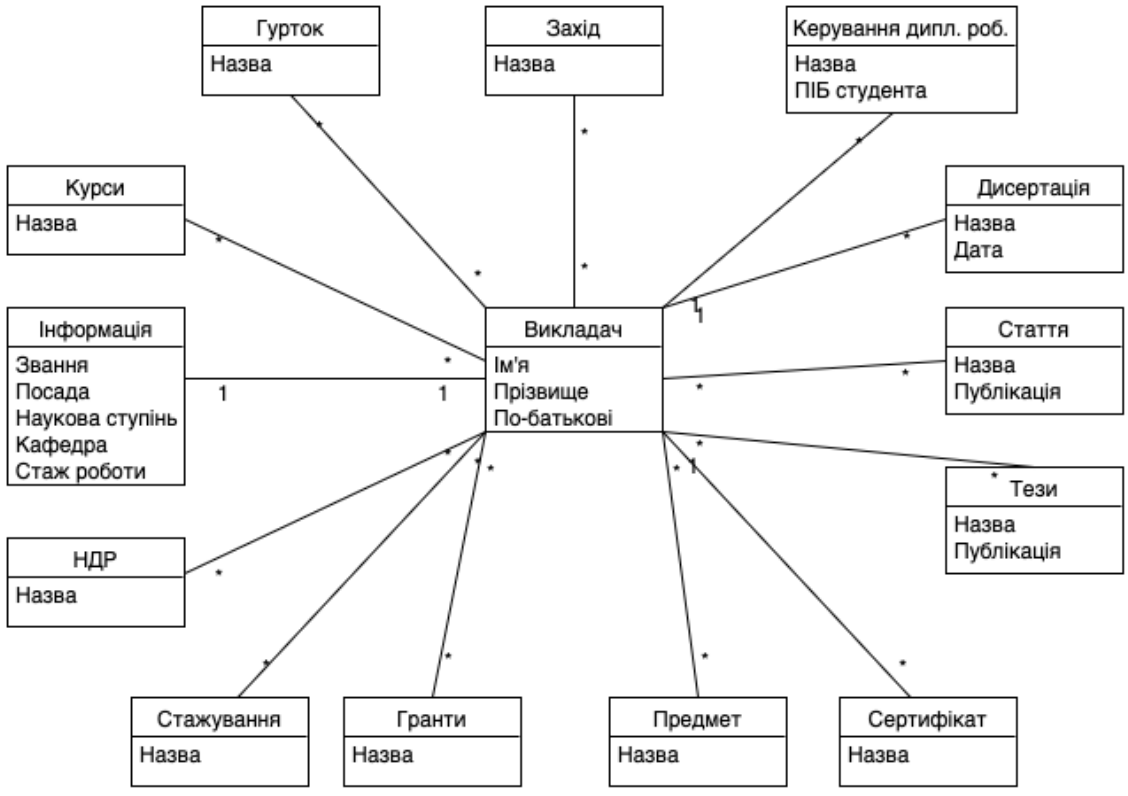
4 РЕАЛІЗАЦІЯ ПРОЕКТУ

Ініціалізація додатку

Функціонал програми включає в себе наступні функції: введення-виведення даних; обробка даних; редагування даних; пошук даних по базі даних; сортування по базі даних; формування звіту. Додаток має відповідати наступним вимогам до надійності: передбачати контроль введеної інформації; передбачати блокування некоректних дій користувача при роботі з системою; забезпечувати цілісність інформації, що зберігається. Програма повинна відповідати наступним вимоги до програмної документації: тексти програми повинні містити всі необхідні коментарі; програмна система повинна включати довідкову інформацію про роботу і підказки користувачеві.

4.1.1 Опис структури зв'язків

Структура зв’язків в додатки будується на ієрархії класів.



Головним класом є Program за допомогою нього виводиться вся інформація додатку.Вся інформація потрапляє до класу Program за допомогою ініціалізації всієї інформації через интерфейс IDBItem

```
IDBItem<Lecturer> dbLecturer = initialization.dbLecturer;
IDBItem<Information> dBInformation = initialization.dBInformation;
IDBItem<ScientificWork> dbScientificWork = initialization.dbScientificWork;
IDBItem<Internship> dbIntership = initialization.dbIntership;
IDBItem<Grants> dbGrants = initialization.dbGrants;
IDBItem<Subject> dbSubject = initialization.dbSubject;
IDBItem<Sertificate> dbSertificate = initialization.dbSertificate;
IDBItem<These> dbThese = initialization.dbThese;
IDBItem<Article> dbArticle = initialization.dbArticle;
IDBItem<Disertation> dbDisertation = initialization.dbDisertation;
IDBItem<ThesisManagement> dbThesisManagment = initialization.dbThesisManagment;
IDBItem<Proposition> dbProposition = initialization.dbProposition;
IDBItem<WorkShop> dbWorkShop = initialization.dbWorkShop;
IDBItem<Course> dbCourse = initialization.dbCourse;
```

та MainMenu ,який запускає програму:

```
MainMenu mainMenu = new MainMenu();
InstanceRun instanceRun = new InstanceRun(mainMenu);
instanceRun.Run();
```

Так як ідея програми заключається в інформації про викладачів то перший клас який потрібно створити - клас Викладачів - Lecturer.

Ініціалізувати всі данні які буде передавати данний клас :

```
public class Lecturer : IID
{
    public int Id { get; set; }
    public string Surname { get; set; }
    public string Name { get; set; }
    public string MiddleName { get; set; }
    public Lecturer()
    {

    }
    public Lecturer(string Surname, string Name, string MiddleName)
    {
        this.Id = Id;
        this.Surname = Surname;
        this.Name = Name;
        this.MiddleName = MiddleName;
    }
    public override string ToString()
    {
        return String.Format(Id + " " + Surname + " " + Name + " " + MiddleName);
    }
```

Для того щоб була можливість пошуку, створення ,видалення,редагування і перегляду треба створити клас вищий в ієрархії -(DataManipulation), і в ньому створити конструктори які ми можемо використати в майбутньому.

```
public class DataManipulation<T> where T:IID
{
    public DBItem<T> dbT { get; set; }
    public DataManipulation()
    {
        this.dbT = DBItem<T>.Instance();
    }
    public void Create(T item)
    {
        dbT.AddItem(item);
    }
    public bool Delete(int id)
    {
        bool isDone = false;
        T item = dbT.FindById(id);
        if (item != null)
        {
            isDone = dbT.Delete(item);
        }
        return isDone;
    }
    public void Update(T itemOld,T itemNew)
    {
        bool isDone = dbT.Delete(itemOld);
        if (isDone == true)
        {
            itemNew.Id = itemOld.Id;
            dbT.UpdateItem(itemNew);
        }
    }
    public T FindById(int id)
    {
        T item = dbT.FindById(id);
        return item;
    }
```

Для того щоб у користувача була можливість пошуку, створення ,видалення,редагування і перегляду Викладачів треба створити клас (DataManipulationLecturer) в якому ми повинні створити конструктори викладачів.

```
public class DataManipulationLecturer : IDataManipulationLecturer
{
    public DBItem<Lecturer> dbLecturer { get; set; }
    public DataManipulationLecturer()
    {
        this.dbLecturer = DBItem<Lecturer>.Instance();
    }
    public void CreateLecturer(string Surname, string Name, string MiddleName)
    {
        Lecturer lecturer = new Lecturer(Surname, Name, MiddleName);
        dbLecturer.AddItem(lecturer);
    }
    public bool DeleteLecturer(int id)
    {
        bool isDone = false;
        Lecturer lecturer = dbLecturer.FindById(id);
        if (lecturer != null)
        {
            isDone = dbLecturer.Delete(lecturer);
        }
        return isDone;
    }
    public void UpdateLecturer(Lecturer lecturer, int id, string Surname, string Name, string MiddleName)
    {
        bool isDone = dbLecturer.Delete(lecturer);
        if (isDone == true)
        {
            Lecturer lecturerNew = new Lecturer(Surname, Name, MiddleName);
            lecturerNew.Id = id;
            dbLecturer.UpdateItem(lecturerNew);
        }
    }
    public Lecturer FindById(int id)
    {
        Lecturer lecturer = dbLecturer.FindById(id);
        return lecturer;
    }
```

Також потрібно додати пошук викладача по прізвищу,імені та по батькові за допомогою класу (Services):

```
public class Services : IServices
{
    IDataManipulationLecturer dataManipulationLecturer;
    public Services()
    {
        dataManipulationLecturer = new DataManipulationLecturer();
    }
    public Lecturer FindLecturerBySurname(string surname)
    {
        Lecturer result = default(Lecturer);
        foreach (Lecturer lecturer in dataManipulationLecturer.dbLecturer.Items)
        {
            if (surname == lecturer.Surname)
            {
                result = lecturer;
            }
        }
        return result;
    }
    public Lecturer FindLecturerByName(string name)
    {
        Lecturer result = default(Lecturer);
        foreach (Lecturer lecturer in dataManipulationLecturer.dbLecturer.Items)
        {
            if (name == lecturer.Name)
            {
                result = lecturer;
            }
        }
    }
    public Lecturer FindLecturerByName(string middleName)
    {
        Lecturer result = default(Lecturer);
        foreach (Lecturer lecturer in dataManipulationLecturer.dbLecturer.Items)
        {
            if (middleName == lecturer.middleName)
            {
                result = lecturer;
            }
        }
    }
}
```

Та створюємо інтерфейс для роботи класу (IServices):

```
public interface IServices
{
    Lecturer FindLecturerBySurname(string surname);
    Lecturer FindLecturerByName(string name);
    Lecturer FindLecturerByMiddleName(string middleName);
}
```

Для того щоб користувачу було легко орієнтуватися в тому що він робить потрібен клас UILecturer ,який створений для комфортної роботи з користувачем :

```
public UILecturer()
{
    dataManipulation = new DataManipulation<Lecturer>();
}
public void Create()
{
    AbstractInstance checkInputString = new CheckInputString("Lecturer create Surname: ", 20);
    string surname = checkInputString.Run();
    string name = new CheckInputString("Lecturer create Name :", 20).Run();
    string middleName = new CheckInputString("Lecturer create MiddleName :", 20).Run();
    Lecturer lecturer = new Lecturer(surname, name, middleName);
    dataManipulation.Create(lecturer);
}
public void Show()
{
    foreach (var item in dataManipulation.dbT.Items)
    {
        Console.WriteLine(item);
    }
}
public void Delete()
{
    int lecturerId = Helper.CheckIntInput(" Delete Lecturer, Lecturer Id : ");
    bool isDone = dataManipulation.Delete(lecturerId);
    if (isDone == true)
    {
        Console.WriteLine("Deleted");
    }
    else
    {
        Console.WriteLine("Not Found");
    }
}
```

```
public void Update()
{
    string surname = "Surname";
    string name = "Name";
    string middleName = "MiddleName";
    int lecturerId = Helper.CheckIntInput("Lecturer Update, Lecturer Id: ");
    Lecturer lecturer = dataManipulation.FindById(lecturerId);
    if (lecturer != null)
    {
        bool isDone = false;
        do
        {
            Console.Write("Lecture surname update?: {0} (y/n): ", lecturer.Surname);
            if ("y" == Console.ReadLine())
            {
                surname = new CheckInputString("Lecturer update, Lecturer Surname :", 20).Run();
            }
            else
            {
                surname = lecturer.Surname;
            }
            Console.Write("Lecture name update?: {0} (y/n): ", lecturer.Name);
            if ("y" == Console.ReadLine())
            {
                name = new CheckInputString("Lecturer update , LecturerName update?:", 20).Run();
            }
            else
            {
                name = lecturer.Name;
            }
            Console.Write("Lecture Middlename update?: {0} (y/n): ", lecturer.MiddleName);
            if ("y" == Console.ReadLine())
            {
                middleName = new CheckInputString("Lecturer update Middlename update? :", 20).Run();
            }
            else
            {
                middleName = lecturer.MiddleName;
            }
            Console.Write("surname: {0} name: {1} middlename: {2} Is Okey? (y/n): ", surname, name, middleName);
            if ("y" == Console.ReadLine())
            {
                isDone = true;
            }
        }
        while (!isDone);
        Lecturer lecturerNew = new Lecturer(surname, name, middleName);
        dataManipulation.Update(lecturer,lecturerNew);
```

Також потрібно створити меню для пошуку, створення ,видалення,редагування і перегляду Викладачів для цього нам потрібно створити головне меню (Main Menu) та меню викладачів (Lecturer Menu):

```
class MainMenu : IInstance
{
    InstanceRun instanceRunLecturer;
    InstanceRun instanceRunInformation;
    InstanceRun instanceRunCourse;
    InstanceRun instanceRunLecturerCourse;
    InstanceRun instanceRunInternship;
    InstanceRun instanceRunWorkShop;
    InstanceRun instanceRunLecturerInternship;
    InstanceRun instanceRunLecturerSubject;

    ConsoleColor colorDefault;
    public bool IsDone { get;set; }
    void IInstance.CleanUp()
    {
        Console.ForegroundColor = colorDefault;
    }

    void IInstance.Idle()
    {
        Console.Clear();
        int number = Helper.CheckIntInput("Menu\n\nSelect Action:\n\n1.Lecturer Menu
        switch (number)
        {
            case 1:
                {
                    instanceRunLecturer.Run();
                    Console.Clear();
                    break;
                }
```

```
public class LecturerMenu:IInstance
{
    UILecturer uiLecturer;
    ConsoleColor colorDefault;
    public bool IsDone { get; set; }
    public void Init()
    {
        colorDefault = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.DarkBlue;
        Console.Title = string.Format("Lecturer Menu");
        uiLecturer = new UILecturer();
    }
    public void Idle()
    {
        Console.Clear();
        int number = Helper.CheckIntInput("Menu Lecturer\n\nSelect Action:\n\n1.Create\n\n2.Show\n\n3.Update\n\n4.Delete\n\n5.Exit\n\n->");
        switch (number)
        {
            case 1:
                {
                    Console.Clear();
                    uiLecturer.Create();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 2:
                {
                    Console.Clear();
                    uiLecturer.Show();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 3:
                {
                    Console.Clear();
                    uiLecturer.Update();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
```

```
            case 4:
                {
                    Console.Clear();
                    uiLecturer.Delete();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 5:
                {
                    Console.Clear();
                    IsDone = true;
                    break;
                }
            default: break;
        }
    }
    public void CleanUp()
    {
        Console.ForegroundColor = colorDefault;
    }
```

Наразі ми можемо запустити програму та побачити головне меню та меню викладачів в якому ми можемо робити всі вище вказані дії.

```
Menu
Select Action:
1.Lecturer Menu
2.Information Menu
3.Course menu
4.Work Shop Menu
5.Internship Menu
6.Lecturer Course Menu
7.Lecturer Internship Menu
8.Lecturer Subject Menu
9.Exit
->
```

```
Menu Lecturer
Select Action:
1.Create
2.Show
3.Update
4.Delete
5.Exit
->
```

Тепер ми маємо змогу створювати нових викладачів ,переглядати їх список ,оновлювати або видаляти їх.

При спробі створити нового викладача ,з'явиться таке вікно із запитаннями :

```
Lecturer create Surname: Прізвище нового викладача
Lecturer create Name :Імя нового викладача
Lecturer create MiddleName :По-батькові нового викладача
```

При спробі подивитися список ,з'явиться вже існуючий список викладачів :

```
1 Vlasenko Oleksiy Ivanovich
2 Fedorenko Olexandr Petrovich
3 Konishevskiy Vladislav Vasylovich
```

При спробі оновити інформацію про вже існуючого викладача ,з'явиться наступне вікно з запитаннями :

```
Lecturer Update, Lecturer Id: 1
Lecture surname update?: Vlasenko (y/n): y
Lecturer update, Lecturer Surname :Нове Прізвище
Lecture name update?: Oleksiy (y/n): n
Lecture Middlename update?: Ivanovich (y/n): n
surname: Нове Прізвище  name: Oleksiy middlename: Ivanovich Is Okey? (y/n): y
```

Перше питання ,це ID викладача про якого ви хочете оновити інформацію ,потім уточнюючі питання що саме ви хочете змінити ,та уточнення внесеної вами нової інформації.

При спробі створити нового викладача ,з'явиться таке вікно із запитанням :

```
 Delete Lecturer, Lecturer Id : 1
Deleted
```

Программа спитає ID викладача ,якого ви хочете видалити .Якщо ви оберете коректний ID то система видалить викладача з бази ,якщо ні ,то система відповість що ID не вірний і спросить ще раз.

Додавання інформації про викладачів .

Дивлячись на діаграму структури зв’язків ми можемо побачити що вся інформація пов'язана з викладачами.Після створення всіх класів інформації про викладача потрібно створити класи які будуть з’єднувати інформацію та викладача .Для цього нам потрібен клас (IID)до якого ми будемо посилатися в класах при створенні,редагуванні,видалені та перегляді інформації про викладача:

```
public interface IID
{
    int Id { get; set; }
}
```

Тепер ми можемо створити класи які будуть об'єднувати потрібні нам класи (LecturerCourse,LecturerInternship,LecturerWorkShop і т.д):

```
public class LecturerCourse:IID
{
    public int Id { get; set; }
    public int LecturerId { get; set; }
    public int CourseId { get; set; }
    public LecturerCourse(int LecturerId, int CourseId)
    {
        this.LecturerId = LecturerId;
        this.CourseId = CourseId;
    }
    public override string ToString()
    {
        return String.Format(Id + " " + LecturerId + " " + CourseId);
    }
}
```

```
public class LecturerInternship:IID
{
    public int Id { get; set; }
    public int LecturerId { get; set; }
    public int InternshipId { get; set; }
    public LecturerInternship(int LecturerId, int InternshipId)
    {
        this.LecturerId = LecturerId;
        this.InternshipId = InternshipId;
    }
    public override string ToString()
    {
        return String.Format(Id + " " + LecturerId + " " + InternshipId);
    }
}
```

```
public class LecturerWorkShop:IID
{
    public int Id { get; set; }
    public int LecturerId { get; set; }
    public int WorkShopId { get; set; }
    public LecturerWorkShop(int LecturerId, int WorkShopId)
    {
        this.LecturerId = LecturerId;
        this.WorkShopId = WorkShopId;
    }
    public override string ToString()
    {
        return String.Format(Id + " " + LecturerId + " " + WorkShopId);
    }
}
```

Також потрібно створити їх меню та додати до класу Program:

```
public class LecturerInternshipMenu:IInstance
{
    UILecturerInternship uiLecturerInternship;
    ConsoleColor colorDefault;
    public bool IsDone { get; set; }
    public void Init()
    {
        colorDefault = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.DarkBlue;
        Console.Title = string.Format("Lecturer Internship Menu");
        uiLecturerInternship = new UILecturerInternship();
    }
    public void Idle()
    {
        Console.Clear();
        int number = Helper.CheckIntInput("Lecturer Internship Menu\n\nSelect Action:\n\n1.Show\n\n2.Exit\n\n->");
        switch (number)
        {
            case 1:
                {
                    Console.Clear();
                    uiLecturerInternship.Show();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 2:
                {
                    Console.Clear();
                    IsDone = true;
                    break;
                }
            default: break;
        }
    }
    public void CleanUp()
    {
        Console.ForegroundColor = colorDefault;
    }
}
```

```
public class LecturerSubjectMenu:IInstance
{
    UILecturerSubject uiLecturerSubject;
    ConsoleColor colorDefault;
    public bool IsDone { get; set; }
    public void Init()
    {
        colorDefault = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.DarkBlue;
        Console.Title = string.Format("Lecturer Subject Menu");
        uiLecturerSubject = new UILecturerSubject();
    }
    public void Idle()
    {
        Console.Clear();
        int number = Helper.CheckIntInput("Lecturer Subject Menu\n\nSelect Action:\n\n1.Show\n\n2.Exit\n\n->");
        switch (number)
        {
            case 1:
                {
                    Console.Clear();
                    uiLecturerSubject.Show();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 2:
                {
                    Console.Clear();
                    IsDone = true;
                    break;
                }
            default: break;
        }
    }
    public void CleanUp()
    {
        Console.ForegroundColor = colorDefault;
    }
}
```

```
public class LecturerCourseMenu:IInstance
{
    UILecturerCourse uiLecturerCourse;
    ConsoleColor colorDefault;
    public bool IsDone { get; set; }
    public void Init()
    {
        colorDefault = Console.ForegroundColor;
        Console.ForegroundColor = ConsoleColor.DarkBlue;
        Console.Title = string.Format("LecturerCourse Menu");
        uiLecturerCourse = new UILecturerCourse();
    }
    public void Idle()
    {
        Console.Clear();
        int number = Helper.CheckIntInput("LecturerCourse Menu\n\nSelect Action:\n\n1.Show\n\n2.Exit\n\n->");
        switch (number)
        {
            case 1:
                {
                    Console.Clear();
                    uiLecturerCourse.Show();
                    Console.ReadKey();
                    Console.Clear();
                    break;
                }
            case 2:
                {
                    Console.Clear();
                    IsDone = true;
                    break;
                }
            default: break;
        }
    }
    public void CleanUp()
    {
        Console.ForegroundColor = colorDefault;
    }
}
```

Для того щоб внести початкову базу створимо клас (Initialization) та внесемо початкову інформацію про викладачів :

```
public void Initial()
{
    dbLecturer = DBItem<Lecturer>.Instance();
    Lecturer lecturerClass1 = new Lecturer("Vlasenko", "Oleksiy", "Ivanovich");
    dbLecturer.AddItem(lecturerClass1);
    Lecturer lecturerClass2 = new Lecturer("Fedorenko", "Olexandr", "Petrovich");
    dbLecturer.AddItem(lecturerClass2);
    Lecturer lecturerClass3 = new Lecturer("Konishevskiy", "Vladislav", "Vasylovich");
    dbLecturer.AddItem(lecturerClass3);

    dBInformation = DBItem<Information>.Instance();
    Information information1 = new Information(1, "Rector", "Rector", "Ph.D", "IT", "Five years");
    dBInformation.AddItem(information1);
    Information information2 = new Information(2, "Deputy rector", "Deputy rector", "PhD", "IT", "Six years");
    dBInformation.AddItem(information2);

    dbScientificWork = DBItem<ScientificWork>.Instance();
    ScientificWork scientificWork1 = new ScientificWork("Name of ScientificWork One");
    dbScientificWork.AddItem(scientificWork1);
    ScientificWork scientificWork2 = new ScientificWork("Name of ScientificWork Two");
    dbScientificWork.AddItem(scientificWork2);

    dbIntership = DBItem<Internship>.Instance();
    Internship internship1 = new Internship("Intership One");
    dbIntership.AddItem(internship1);
    Internship internship2 = new Internship("Intership Two");
    dbIntership.AddItem(internship2);

    dbGrants = DBItem<Grants>.Instance();
    Grants grants1 = new Grants("GrantOne");
    dbGrants.AddItem(grants1);
    Grants grants2 = new Grants("GrantTwo");
    dbGrants.AddItem(grants2);

    dbSubject = DBItem<Subject>.Instance();
    Subject subject1 = new Subject("English");
    dbSubject.AddItem(subject1);
    Subject subject2 = new Subject("Programming");
    dbSubject.AddItem(subject2);

    dbSertificate = DBItem<Sertificate>.Instance();
    Sertificate sertificate1 = new Sertificate("Sertificate One");
    dbSertificate.AddItem(sertificate1);
    Sertificate sertificate2 = new Sertificate("Sertificate Two");
    dbSertificate.AddItem(sertificate2);
```

```
dbThese = DBItem<These>.Instance();
These these1 = new These("TheseOne", "Publication One");
dbThese.AddItem(these1);
These these2 = new These("TheseTwo", "Publication Two");
dbThese.AddItem(these2);

dbArticle = DBItem<Article>.Instance();
Article article1 = new Article("ArticleOne", "Publication One");
dbArticle.AddItem(article1);
Article article2 = new Article("ArticleTwo", "Publication Two");
dbArticle.AddItem(article1);

dbDisertation = DBItem<Disertation>.Instance();
Disertation disertation1 = new Disertation(01, 01, 2003);
dbDisertation.AddItem(disertation1);
Disertation disertation2 = new Disertation(02, 02, 2004);
dbDisertation.AddItem(disertation2);

dbThesisManagment = DBItem<ThesisManagement>.Instance();
ThesisManagement thesisManagement1 = new ThesisManagement("First Title Thesis Management", "Bondar", "Viktor", "Andreevich");
dbThesisManagment.AddItem(thesisManagement1);
ThesisManagement thesisManagement2 = new ThesisManagement("Second Title Thesis Management", "Karpovets", "Anastasia", "Viktorovna");
dbThesisManagment.AddItem(thesisManagement2);

dbProposition = DBItem<Proposition>.Instance();
Proposition proposition1 = new Proposition("First Title Proposition ");
dbProposition.AddItem(proposition1);
Proposition proposition2 = new Proposition("Second Title Proposition ");
dbProposition.AddItem(proposition2);

dbWorkShop = DBItem<WorkShop>.Instance();
WorkShop workShop1 = new WorkShop("First Title Work Shop One");
dbWorkShop.AddItem(workShop1);
WorkShop workShop2 = new WorkShop("Second Title Work Shop Two");
dbWorkShop.AddItem(workShop2);

dbCourse = DBItem<Course>.Instance();
Course course1 = new Course("First Title Course One");
dbCourse.AddItem(course1);
Course course2 = new Course("Second Title Course Two");
dbCourse.AddItem(course2);
```

4.3.1

Останнім кроком для роботи програми треба створити класи інтерфейсів для кожного з класів (DataManipulationLecturer, DataManipulationCourse, DataManipulationInternship , DataManipulationWorkShop и т.д)

```
public interface IDataManipulationCourse
{
    IDBItem<Course> dbCourse { get; set; }
}
```

```
public interface IDataManipulationWorkShop
{
    IDBItem<WorkShop> dbWorkShop { get; set; }
}
```

```
public interface IDataManipulationInternship
{
    IDBItem<Internship> dbInternship { get; set; }
}
```

Наразі при запуску програми ми будемо бачити головне меню і матимемо змогу знайти інформацію про кожного створеного викладача ,або створити нового і додати інформацію.

```
Menu Information
Select Action:
1.Create
2.Show
3.Update
4.Delete
5.Exit
->
```

В Menu Information ми зможемо побачити інформацію про викладачів та створити або оновити її

```
Course Menu
Select Action:
1.Create
2.Show
3.Update
4.Delete
5.Exit
->
```

В Menu Course ми зможемо побачити курси які проводить викладачі а також створити нові курси або оновити їх

```
Work Shop Menu
Select Action:
1.Create
2.Show
3.Update
4.Delete
5.Exit
->
```

В Menu Work Shop ми можемо побачити гуртки які проводить викладачі а також створити нові курси або оновити їх

```
LecturerCourse Menu

Select Action:

1.Show

2.Exit

->
```

В Menu Lecturer Course ми можемо подивитися які курси проводить який викладач.

Реалізація основних елементів додатка

Для введення і виведення даних використовуються компоненти: Initialization,Program,MainMenu.

```
dbLecturer = DBItem<Lecturer>.Instance();
Lecturer lecturerClass1 = new Lecturer("Vlasenko", "Oleksiy", "Ivanovich");
dbLecturer.AddItem(lecturerClass1);
Lecturer lecturerClass2 = new Lecturer("Fedorenko", "Olexandr", "Petrovich");
dbLecturer.AddItem(lecturerClass2);
Lecturer lecturerClass3 = new Lecturer("Konishevskiy", "Vladislav", "Vasylovich");
dbLecturer.AddItem(lecturerClass3);

dBInformation = DBItem<Information>.Instance();
Information information1 = new Information(1, "Rector", "Rector", "Ph.D", "IT", "Five years");
dBInformation.AddItem(information1);
Information information2 = new Information(2, "Deputy rector", "Deputy rector", "PhD", "IT", "Six years");
dBInformation.AddItem(information2);

dbScientificWork = DBItem<ScientificWork>.Instance();
ScientificWork scientificWork1 = new ScientificWork("Name of ScientificWork One");
dbScientificWork.AddItem(scientificWork1);
ScientificWork scientificWork2 = new ScientificWork("Name of ScientificWork Two");
dbScientificWork.AddItem(scientificWork2);

dbIntership = DBItem<Internship>.Instance();
Internship internship1 = new Internship("Intership One");
dbIntership.AddItem(internship1);
Internship internship2 = new Internship("Intership Two");
dbIntership.AddItem(internship2);

dbGrants = DBItem<Grants>.Instance();
Grants grants1 = new Grants("GrantOne");
dbGrants.AddItem(grants1);
Grants grants2 = new Grants("GrantTwo");
dbGrants.AddItem(grants2);

dbSubject = DBItem<Subject>.Instance();
Subject subject1 = new Subject("English");
dbSubject.AddItem(subject1);
Subject subject2 = new Subject("Programming");
dbSubject.AddItem(subject2);

dbSertificate = DBItem<Sertificate>.Instance();
Sertificate sertificate1 = new Sertificate("Sertificate One");
dbSertificate.AddItem(sertificate1);
Sertificate sertificate2 = new Sertificate("Sertificate Two");
dbSertificate.AddItem(sertificate2);
MainMenu mainMenu = new MainMenu();
InstanceRun instanceRun = new InstanceRun(mainMenu);
instanceRun.Run();
```

```
int number = Helper.CheckIntInput("Menu\n\nSelect Action:\n\n1.Lecturer Menu\n\n2.Information Menu\n\n3.Course menu\n\n4.Work Shop !
switch (number)
{
    case 1:
        {
            instanceRunLecturer.Run();
            Console.Clear();
            break;
        }
    case 2:
        {
            instanceRunInformation.Run();
            Console.Clear();
            break;
        }
    case 3:
        {
            instanceRunCourse.Run();
            Console.Clear();
            break;
        }
    case 4:
        {
            instanceRunWorkShop.Run();
            Console.Clear();
            break;
        }
    case 5:
        {
            instanceRunInternship.Run();
            Console.Clear();
            break;
        }
    case 6:
        {
            instanceRunLecturerCourse.Run();
            Console.Clear();
            break;
        }
    case 7:
        {
            instanceRunLecturerInternship.Run();
            Console.Clear();
```

Кнопка «Додати»додає новий запис за допомогою класу DBItem:

```
public class DBItem<T> : IDBItem<T> where T : IID
{
    private static DBItem<T> instance = null;
    public static DBItem<T> Instance()
    {
        if (instance == null)
        {
            instance = new DBItem<T>();
        }
        return instance;
    }
    int count = 1;
    public List<T> Items { get; set; }
    private DBItem()
    {
        this.Items = new List<T>();
    }
    public void AddItem(T item)
    {
        item.Id = count++;
        Items.Add(item);
```

Кнопка «Змінити» змінює запис, за допомогою класу DBItem

```
public void UpdateItem(T item)
{
    Items.Add(item);
}
```

Кнопка «Видалити» видаляє запис таблиці,за допомогою класу DBItem.

```
public bool Delete(T item)
{
    return Items.Remove(item);
}
```

5 ВИСНОВКИ

В результаті виконання даної роботи був розроблений додаток “Lecturer Online ” на фреймворку .NET.Core. Також були вирішені такі задачі:

1. Проведено аналіз існуючих рішень розробки додатків, їх переваги та недоліки. На основі аналізу визначено що розробка на фреймворку .NET.Core має багато переваг перед своїми конкурентами та є оптимальним рішенням для розробки додатку.
2. Визначено вимоги та функціональні можливості додатку “Lecturer Online”.
3. Був обраний оптимальний стек технологій.
4. Ініціалізація проекту, установка та налаштування обраних бібліотек. Створення та налаштування бази даних.Створення діаграм реалізації та варіантів використання.Розроблена архітектура проекту.

Перспективи проекту. Планується покращення працездатності розробленого додатку. Заповнення додатку новим функціоналом.Та покращення роботи БД.

6 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1) Glossary tech [Електронний ресурс] – Режим доступу : https://glossarytech.com (дата звернення 21.04.2021). - Назва з екрана.
2) Csharp [Електронний ресурс] – Режим доступу : http://csharp.net-informations.com (дата звернення 21.04.2021). - Назва з екрана.
3) GeeksForGeeks [Електронний ресурс] – Режим доступу : https://www.geeksforgeeks.org (дата звернення 21.04.2021). - Назва з екрана.
4) ProfessorWeb [Електронний ресурс] – Режим доступу : https://professorweb.ru (дата звернення 21.04.2021). - Назва з екрана.
5) ItProger [Електронний ресурс] – Режим доступу : https://itproger.com (дата звернення 21.04.2021). - Назва з екрана.
6) Microsoft.docs [Електронний ресурс] – Режим доступу : https://docs.microsoft.com (дата звернення 21.04.2021). - Назва з екрана.
7) C-sharp [Електронний ресурс] – Режим доступу : https://c-sharp.pro (дата звернення 21.04.2021). - Назва з екрана
8) Habr [Електронний ресурс] – Режим доступу : https://habr.com (дата звернення 21.04.2021). - Назва з екрана
9) Coding-Geek [Електронний ресурс] – Режим доступу : http://coding-geek.com (дата звернення 21.04.2021). - Назва з екрана
10) Quora [Електронний ресурс] – Режим доступу : quora.com (дата звернення 21.04.2021). - Назва з екрана
11) Oracle [Електронний ресурс] – Режим доступу : https://www.oracle.com (дата звернення 21.04.2021). - Назва з екрана
12) Selectel [Електронний ресурс] – Режим доступу :www.selectel.ru (дата звернення 21.04.2021). - Назва з екрана
13) Support.Microsoft [Електронний ресурс] – Режим доступу : support.microsoft.com (дата звернення 21.04.2021). - Назва з екрана
14) Lessons-tva [Електронний ресурс] – Режим доступу : lessons-tva.info (дата звернення 21.04.2021). - Назва з екрана
15) Proglib [Електронний ресурс] – Режим доступу : proglib.io (дата звернення 21.04.2021). - Назва з екрана
16) QnaHabr [Електронний ресурс] – Режим доступу : qna.habr.com (дата звернення 21.04.2021). - Назва з екрана
17) Metanin [Електронний ресурс] – Режим доступу : metanit.com (дата звернення 21.04.2021). - Назва з екрана

18) Specialist [Електронний ресурс] – Режим доступу : specialist.ru (дата звернення 21.04.2021). - Назва з екрана
19) Code-live [Електронний ресурс] – Режим доступу : code-live.ru (дата звернення 21.04.2021). - Назва з екрана